

## Objectif :

Le but de ce workshop est d'installer le package « Api-Platform » en utilisant Symfony flex.

### 1) Définition d'Api-Platform :



Api-Platform est une distribution Symfony permettant de générer des services REST pour des entités Doctrine. Créé par Kévin Dunglas (contributeur Core Symfony et fondateur des Tilleuls.coop), il est maintenant développé et maintenu par de grands noms de la communauté et fait partie des distributions officielles de Symfony (maintenant installées via l'outil Flex).

### 2) Etapes d'installation :

Symfony propose deux versions :

- La version **skeleton**, la plus minimaliste et légère, dédiée aux applications ayant l'architecture micro-services, qui n'installe que le strict minimum, vous laissant ainsi la liberté d'ajouter les composants dont vous avez réellement besoin.
- La version **webapp**, qui va installer tout le nécessaire pour faire fonctionner une application Web (vues, annotations, base de données, ...)

Dans notre workshop nous allons utiliser la version **webapp**.

- a) Installer la version website-skeleton en utilisant Composer Flex via la commande suivante :

```
symfony new my_project_directory --version=5.4 --webapp
```

La recette symfony/webapp est également disponible sur Github:

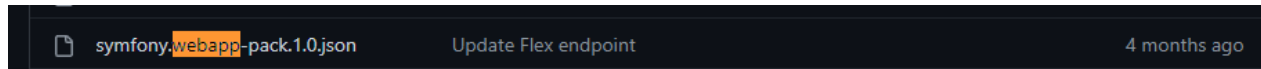
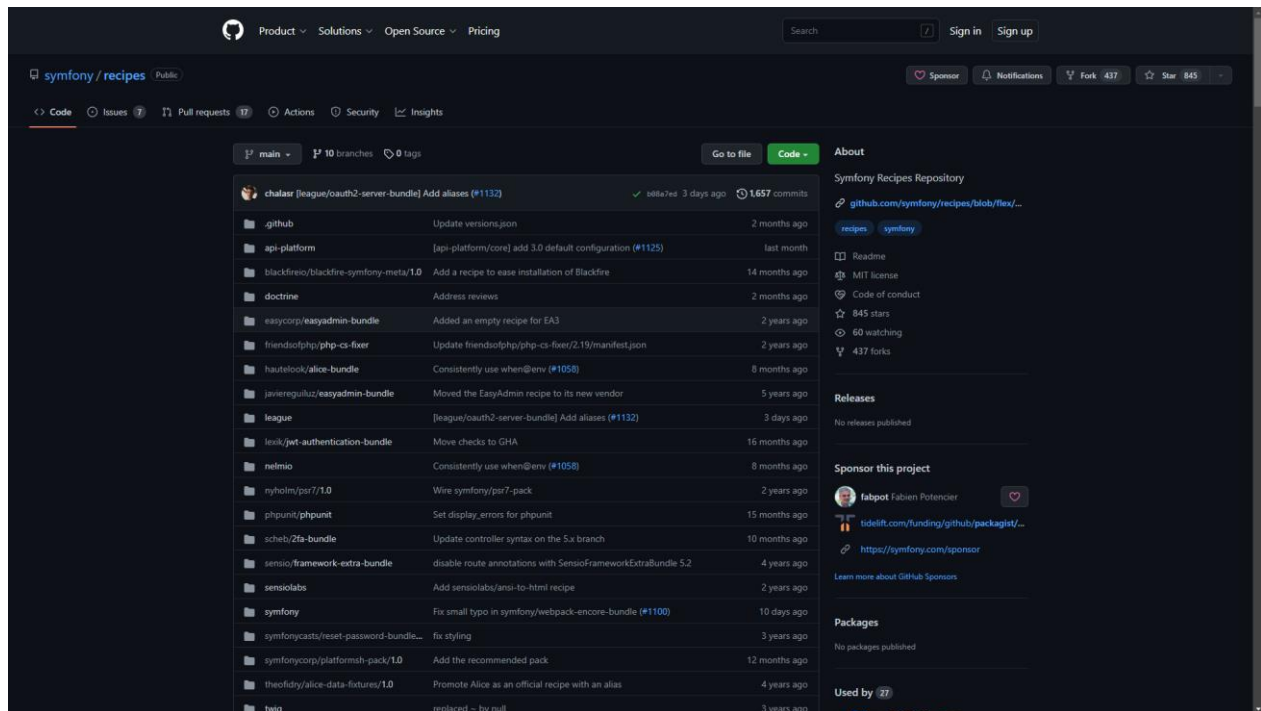


Figure 1: La recette webapp

- b) Nous souhaitons exposer une couche API dans notre future application, nous installerons pour cela API Platform.

-Cherchez le package ou bundle qui fera l'affaire en accédant à

<https://github.com/symfony/recipes> :



-Choisissez la recette api-platform comme illustré dans la figure ci-dessous :

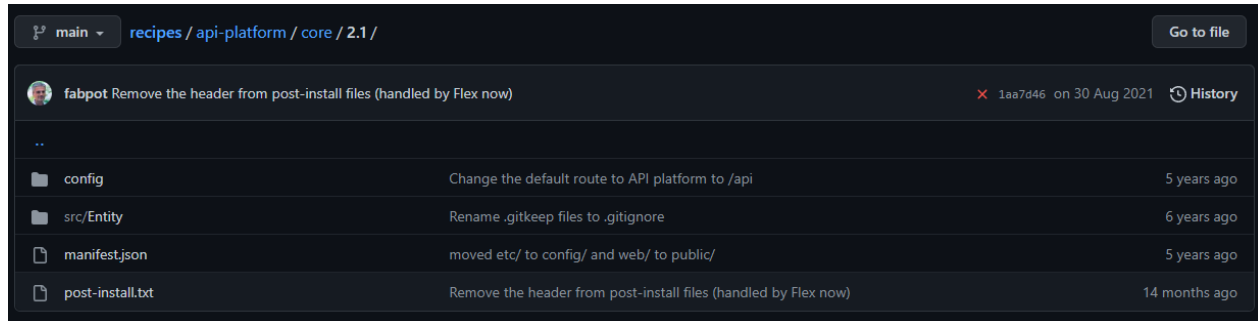


Figure 2: La recette api-platform

### Remarque :

Dans la figure 5, on remarque qu'une recette définit plusieurs informations, la première étant les alias (**Aliases**).

Le package api ou api-platform n'existe pas pour Composer. Mais **Flex** intervient pour détecter si le nom de package demandé correspond à un alias, et si c'est le cas, fait en sorte que Composer installe le paquet correspondant. Donc pour installer un package **api-platform** on peut également écrire directement son alias : api ou api-platform comme suit :

Version sans « Flex »

```
composer require api-platform/api-pack
```

1er Version avec Flex

```
composer require api
```

2ème version avec Flex

```
composer require api-platform
```

Pour avoir plus de détails sur ce package on peut accéder à ce lien:  
<https://packagist.org/packages/api-platform/api-pack>.

**Packagist** The PHP Package Repository

Search packages...

## api-platform/api-pack

composer require api-platform/api-pack

A pack for API Platform

**Maintainers**

**Details**

github.com/api-platform/api-pack

Source

Issues

Installs: 3 216 108

Dependencies: 62

Suggesters: 3

Security: 0

Stars: 508

Watchers: 7

Forks: 29

Open issues: 0

Type: symfony-pack

**v1.3.0** 2020-08-28 20:27 UTC

requires	requires (dev)	suggests
<ul style="list-style-type: none"> <li>api-platform/core: *</li> <li>nelmio/cors-bundle: *</li> <li>symfony/asset: *</li> <li>symfony/expression-language: *</li> <li>symfony/form: *</li> <li>symfony/security-bundle: *</li> <li>symfony/serializer-pack: *</li> <li>symfony/twig-bundle: *</li> <li>symfony/validator: *</li> </ul>	None	None

**dev-master**

requires	requires (dev)	suggests
<ul style="list-style-type: none"> <li>api-platform/core: *</li> <li>nelmio/cors-bundle: *</li> <li>symfony/asset: *</li> <li>symfony/expression-language: *</li> <li>symfony/form: *</li> <li>symfony/security-bundle: *</li> <li>symfony/serializer-pack: *</li> <li>symfony/twig-bundle: *</li> <li>symfony/validator: *</li> </ul>	None	None

**provides**

None

**conflicts**

None

**replaces**

None

MIT [0fb12343362f56565eb374d3c49bec580ffc8d](#)

**README**

This is a Flex pack to install API Platform, a next-generation web framework designed to easily create API-first projects without compromising extensibility and flexibility.

Installation instructions are available in the [API Platform documentation](#).

**License**

The MIT License (MIT). Please see [License File](#) for more information.

**About Packagist**

Atom/RSS Feeds

**Statistics**

Browse Packages

**API**

Mirrors

Packagist maintenance and hosting is provided by [Private Packagist](#)

-Lancez la commande suivante :

*composer require api*

Vous allez remarquer dans votre console la trace de l'exécution de cette commande comme présenté dans la figure suivante :

```

TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

$ composer require api
Using version ^1.3 for api-platform/api-pack
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Restricting packages listed in "symfony/symfony" to "5.2.*"
Package operations: 3 installs, 0 updates, 0 removals
  - Installing symfony/serializer-pack (v1.0.4): Loading from cache
  - Installing symfony/orm-pack (v2.1.0): Loading from cache
  - Installing api-platform/api-pack (v1.3.0): Loading from cache
Writing lock file
Generating optimized autoload files
composer/package-versions-deprecated: Generating version class...
composer/package-versions-deprecated: ...done generating version class
65 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

Run composer recipes at any time to see the status of your Symfony recipes.

Executing script cache:clear [OK]
Executing script assets:install public [OK]

Unpacked api-platform/api-pack dependencies
Unpacked symfony/orm-pack dependencies
Unpacked symfony/serializer-pack dependencies
Loading composer repositories with package information
Installing dependencies (including require-dev) from lock file
Package operations: 0 installs, 0 updates, 3 removals
  - Removing symfony/serializer-pack (v1.0.4)
  - Removing symfony/orm-pack (v2.1.0)
  - Removing api-platform/api-pack (v1.3.0)
63 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

```

Figure 3: Installation de package "Api-platform" via composer dans le projet Symfony

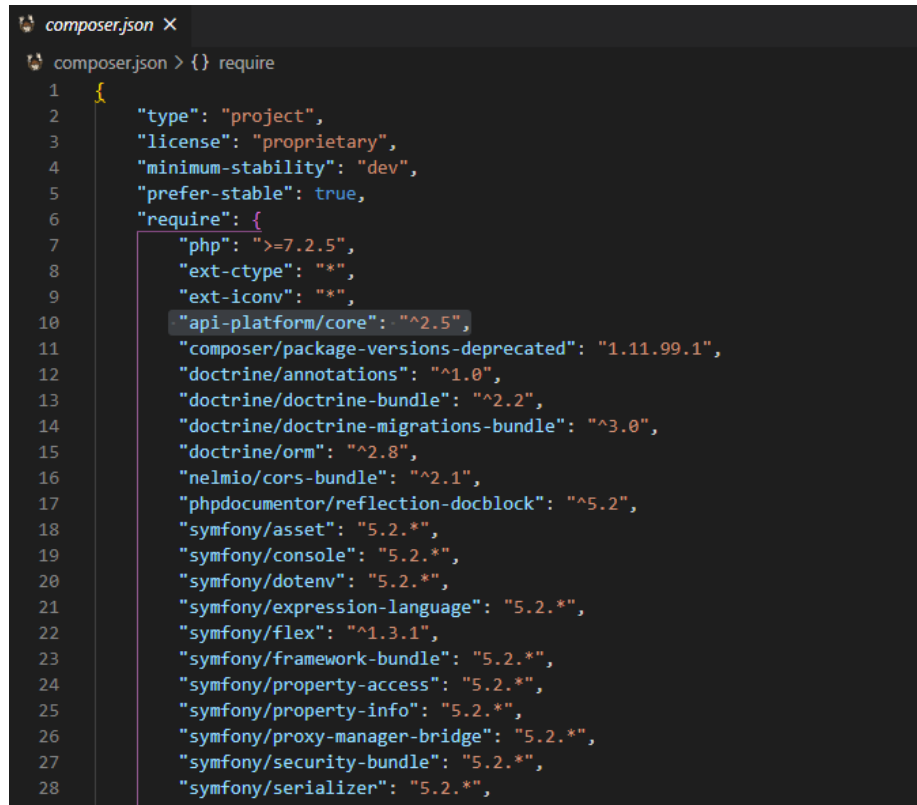
L'application adresse une demande au serveur de Symfony Flex avant d'essayer d'installer le package avec composer. S'il n'y a aucune information sur le paquet que nous voulons installer, le serveur Flex ne renvoie rien et l'installation du paquet suit la procédure habituelle basée sur Composer. S'il y a des informations, Flex les renvoie dans un fichier appelé "recette" (recipes) et l'application l'utilise pour décider quels packages sont à installer et quelles tâches sont à exécuter automatiquement suite à l'installation.

-Après installation, vérifiez le contenu des fichiers suivants :

1. Composer.json

2. Composer.lock
3. Config/bundles.php
4. Symfony.lock

Le contenu sera semblable à celui dans les figures suivantes :



```

1  {
2      "type": "project",
3      "license": "proprietary",
4      "minimum-stability": "dev",
5      "prefer-stable": true,
6      "require": {
7          "php": ">=7.2.5",
8          "ext-ctype": "*",
9          "ext-iconv": "*",
10         "api-platform/core": "^2.5",
11         "composer/package-versions-deprecated": "1.11.99.1",
12         "doctrine/annotations": "^1.0",
13         "doctrine/doctrine-bundle": "^2.2",
14         "doctrine/doctrine-migrations-bundle": "^3.0",
15         "doctrine/orm": "^2.8",
16         "nelmio/cors-bundle": "^2.1",
17         "phpdocumentor/reflection-docblock": "^5.2",
18         "symfony/asset": "5.2.*",
19         "symfony/console": "5.2.*",
20         "symfony/dotenv": "5.2.*",
21         "symfony/expression-language": "5.2.*",
22         "symfony/flex": "^1.3.1",
23         "symfony/framework-bundle": "5.2.*",
24         "symfony/property-access": "5.2.*",
25         "symfony/property-info": "5.2.*",
26         "symfony/proxy-manager-bridge": "5.2.*",
27         "symfony/security-bundle": "5.2.*",
28         "symfony/serializer": "5.2.*",

```

Figure 4: Le contenu du fichier "Composer.json" est mis à jour

```
composer.lock X
composer.lock > [ ] packages > { } 0
1 {
2   "_readme": [
3     "This file locks the dependencies of your project to a known state",
4     "Read more about it at https://getcomposer.org/doc/01-basic-usage.md#installing-dependencies",
5     "This file is @generated automatically"
6   ],
7   "content-hash": "a531ca9341cae3d14b70c9d19893ec68",
8   "packages": [
9     {
10      "name": "api-platform/core",
11      "version": "v2.5.9",
12      "source": {
13        "type": "git",
14        "url": "https://github.com/api-platform/core.git",
15        "reference": "4f05477b5ec5cff7c9324a7510900db2a1173d70"
16      },
17      "dist": {
18        "type": "zip",
19        "url": "https://api.github.com/repos/api-platform/core/zipball/4f05477b5ec5cff7c9324a7510900db2a1173d70",
20        "reference": "4f05477b5ec5cff7c9324a7510900db2a1173d70",
21        "shasum": ""
22      },
23      "require": {
24        "doctrine/inflector": "^1.0 || ^2.0",
25        "fig/link-util": "^1.0",
26        "php": ">=7.1",
27        "psr/cache": "^1.0",
28        "psr/container": "^1.0",
29        "symfony/http-foundation": "^4.4 || ^5.1",
30        "symfony/http-kernel": "^4.4 || ^5.1",
31        "symfony/property-access": "^3.4.19 || ^4.4 || ^5.1",
```

Figure 5: Le contenu du fichier "composer.lock" est mis à jour

```
bundles.php X
config > bundles.php
1 <?php
2
3 return [
4     Symfony\Bundle\FrameworkBundle\FrameworkBundle::class => ['all' => true],
5     Symfony\Bundle\TwigBundle\TwigBundle::class => ['all' => true],
6     Symfony\Bundle\SecurityBundle\SecurityBundle::class => ['all' => true],
7     Doctrine\Bundle\DoctrineBundle\DoctrineBundle::class => ['all' => true],
8     Doctrine\Bundle\MigrationsBundle\Doctrine\MigrationsBundle::class => ['all' => true],
9     Nelmio\CorsBundle\NelmioCorsBundle::class => ['all' => true],
10    ApiPlatform\Core\Bridge\Symfony\Bundle\ApiPlatformBundle::class => ['all' => true],
11 ];
12
```

Figure 6 : Le contenu du fichier "bundles.php" est mis à jour

### Remarque :

Dans les versions de symfony inférieure à la version 4.0, quand il n'y avait pas Flex, le développeur devrait lire la documentation pour savoir comment activer son bundle et pour quel

environnement. Avec Flex, vous allez remarquer que cette ligne a été générée automatiquement.

Flex conserve aussi la trace des recettes qu'il a installées dans un fichier *symfony.lock*, qui doit être validé dans notre référentiel de code.

```

1  {
2      "api-platform/api-pack": {
3          "version": "v1.3.0"
4      },
5      "api-platform/core": {
6          "version": "2.5",
7          "recipe": {
8              "repo": "github.com/symfony/recipes",
9              "branch": "master",
10             "version": "2.5",
11             "ref": "a93061567140e386f107be75340ac2aee3f86cbf"
12         },
13         "files": [
14             "./config/packages/api_platform.yaml",
15             "./config/routes/api_platform.yaml",
16             "./src/Entity/.gitignore"
17         ]
18     },
19     "composer/package-versions-deprecated": {
20         "version": "1.11.99.1"
21     },
22     "doctrine/annotations": {
23         "version": "1.0",
24         "recipe": {
25             "repo": "github.com/symfony/recipes",
26             "branch": "master",
27             "version": "1.0",
28             "ref": "a2759dd6123694c8d901d0ec80006e044c2e6457"
29         }
30     }
31 }
```

Figure 7: Le contenu de fichier "Symfony.lock" est mis à jour

Mais ce qui fait réellement l'intérêt des recettes, c'est la partie **configurator** : qui permet de configurer une partie de votre application. Il existe actuellement 8 types de configurateurs disponibles pour les recettes Flex comme suit :



- **bundles** permet de connaître les bundles à instancier suivant l'environnement ;
- **container** donne les paramètres à ajouter dans votre DIC ;
- **copy-from-package** indique les fichiers à copier du paquet Composer vers votre application Symfony ;
- **copy-from-recipe** indique les fichiers à copier de la recette vers votre application Symfony (y il faut copier le contenu du dossier config/ provenant du recipe dans le dossier de configuration de l'application Symfony).
- **env** donne les variables d'environnement à ajouter dans les fichiers .env.dist et .env de l'application ;
- **composer-scripts** liste les commandes à ajouter dans la section scripts de votre composer.json pour qu'elles soient lancées à la fin des commandes install et update ;
- **gitignore** donne les entrées à ajouter à votre .gitignore ;
- **post-install-output** permet d'afficher du texte lorsque l'installation par Composer se termine.

```

9 lines (9 sloc) | 206 Bytes
1  {
2      "bundles": {
3          "ApiPlatform\\Core\\Bridge\\Symfony\\Bundle\\ApiPlatformBundle": ["all"]
4      },
5      "copy-from-recipe": {
6          "config/": "%CONFIG_DIR%",
7          "src/": "%SRC_DIR%"
8      }
9  }
```

Figure 8: Exemple d'un configurateur de la recette api-platform/core (manifest.json)

Flex est suffisamment intelligent pour également supprimer ce qui a été ajouté par une recette lorsque vous supprimez le paquet associé.

#### Référence :

- <https://github.com/symfony/flex>

- <https://api-platform.com/docs/distribution/#using-symfony-flex-and-composer-advanced-users>
- <https://symfony.com/>
- <https://github.com/symfony/recipes/blob/master/api-platform/core/2.1/manifest.json>